```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

FILEID**RMOWILD

```
RRRRRRRR    MM      MM    OOOOOO    WW      WW    IIIIII    LL          DDDDDDDD
RRRRRRRR    MM      MM    OOOOOO    WW      WW    IIIIII    LL          DDDDDDDD
RR      RR  MMMM  MMMM  OO      OO  WW      WW      II      LL          DD      DD
RR      RR  MMMM  MMMM  OO      OO  WW      WW      II      LL          DD      DD
RR      RR  MM  MM  MM  OO    OOOO  WW      WW      II      LL          DD      DD
RR      RR  MM  MM  MM  OO    OOOO  WW      WW      II      LL          DD      DD
RRRRRRRR    MM      MM  OO  OO  OO  WW      WW      II      LL          DD      DD
RRRRRRRR    MM      MM  OO  OO  OO  WW      WW      II      LL          DD      DD
RR  RR      MM      MM  OOOO    OO  WW  WW  WW      II      LL          DD      DD
RR  RR      MM      MM  OOOO    OO  WW  WW  WW      II      LL          DD      DD
RR    RR    MM      MM  OO      OO  WWWW  WWWW      II      LL          DD      DD      ....
RR    RR    MM      MM  OO      OO  WWWW  WWWW      II      LL          DD      DD      ....
RR      RR  MM      MM    OOOOOO    WW      WW    IIIIII    LLLLLLLLLL  DDDDDDDD        ....
RR      RR  MM      MM    OOOOOO    WW      WW    IIIIII    LLLLLLLLLL  DDDDDDDD        ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          $BEGIN  RMOWILD,000,RMSRMSFILENAME,<DIRECTORY WILDCARDING>,<PIC,NOWRT>
0000     2
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
```

```
0000     28 ;++
0000     29 ;
0000     30 ; Facility: rms32
0000     31 ;
0000     32 ; Abstract:
0000     33 ;       this module contains all the routines
0000     34 ;       necessary to perform directory wildcarding
0000     35 ;       in file specification processing.
0000     36 ;
0000     37 ; Environment:
0000     38 ;       vax/vms
0000     39 ;
0000     40 ; Author:
0000     41 ;       Tim Halvorsen   AUG-1979
0000     42 ;
0000     43 ; Modified By:
0000     44 ;
0000     45 ;       V03-013 DGB0020         Donald G. Blair         06-Mar-1984
0000     46 ;               Use full-length fib in order to support access mode
0000     47 ;               protected files.
0000     48 ;
0000     49 ;       V03-012 RAS0252         Ron Schaefer            13-Feb-1984
0000     50 ;               Fix infinite loop caused by calling FMG$MATCH_NAME
0000     51 ;               with a bad descriptor.  Also clean-up a couple of
0000     52 ;               error paths to prevent looping and setup SWB correctly.
0000     53 ;
0000     54 ;       V03-011 RAS0232         Ron Schaefer            9-Jan-1984
0000     55 ;               Have RM$NEXTDIR return an error if the SWB isn't there.
0000     56 ;               The new error is RMS$_NOVALPRS.  This is a consequence
0000     57 ;               of RAS0219.  Could also be used to detect wild-card
0000     58 ;               directory processing without saved context.
0000     59 ;
0000     60 ;       V03-010 RAS0219         Ron Schaefer            8-Dec-1983
0000     61 ;               Change fwa references to FWA$T_SWB to be a separate
0000     62 ;               structure; use additional SWB fields as temp storage
0000     63 ;               and not random FWA cells.
0000     64 ;
0000     65 ;       V03-009 RAS0192         Ron Schaefer            13-Sep-1983
0000     66 ;               Fix bug in parsing UIC-format wildcard directories of
0000     67 ;               rooted devices.  Problem is caused by using the high
0000     68 ;               word of the directory descriptors to contain the
0000     69 ;               group-member flag.   Also fix wrong register on
0000     70 ;               error path.
0000     71 ;
0000     72 ;       V03-008 KBT0583         Keith B. Thompson       12-Aug1983
0000     73 ;               Clean up some fwa constants
0000     74 ;
0000     75 ;       V03-007 KBT0558         Keith B. Thompson       20-Jul-1983
0000     76 ;               Fix a bug introduced in KBT0512
0000     77 ;
0000     78 ;       V03-006 KBT0532         Keith B. Thompson       1-Jun-1983
0000     79 ;               Make RM$INIT_SWB save r8, remove RM$SKIP_SUBTREE and
0000     80 ;               make the SWA defined in fwadef
0000     81 ;
0000     82 ;       V03-005 KBT0524         Keith B. Thompson       23-May-1983
0000     83 ;               Place this beast in with xpfn
0000     84 ;
```

```
0000     85 ;       V03-004 KBT0512         Keith B. Thompson       17-May-1983
0000     86 ;               Make some rooted directory changes and save the fwa
0000     87 ;               descriptor flags.
0000     88 ;
0000     89 ;       V03-003 KBT0490         Keith B. Thompson       10-Feb-1983
0000     90 ;               Temporary fix so that the swb is the correct size
0000     91 ;
0000     92 ;       V03-002 KBT0474         Keith B. Thompson       26-Jan-1983
0000     93 ;               Fix a bug when parsing [...]*.*;* when in the last sub-
0000     94 ;               directory by putting in a check the first time through
0000     95 ;               in nextdir to see if we are all the way down.
0000     96 ;
0000     97 ;       V03-001 KBT0218         Keith B. Thompson       23-Aug-1982
0000     98 ;               Reorganize psects
0000     99 ;
0000    100 ;       V02-022 KEK0018         K. E. Kinnear           9-Feb-1982
0000    101 ;               Fix ASSUME for SWB offset into translation buffers for
0000    102 ;               new length of FWA$T_NAMEBUF.
0000    103 ;
0000    104 ;       V02-021 JWH0001         Jeffrey W. Horn         18-Jan-1982
0000    105 ;               Fix broken subroutine branch.
0000    106 ;
0000    107 ;       V02-020 TMK0025         Todd M. Katz            16-Dec-1981
0000    108 ;               Rip out all SDI stuff in RM$NEXTDIR. RMOWILD will not
0000    109 ;               (better not) be called for anything but files on disks.
0000    110 ;               Also FWA$B_DIRWCFLGS is being set incorrectly on repeated
0000    111 ;               searches involving ellipsis traversal because it is not
0000    112 ;               being set to the value it had after parse before the match
0000    113 ;               routine is called. Fix this bug by resetting FWA$B_DIRWCFLGS
0000    114 ;               to the value it had (before the first search began) before
0000    115 ;               the match attempt is made. This will mean no change in it
0000    116 ;               if there were no ellipsis, but if there were it will end up
0000    117 ;               being set to its new value on a successful match.
0000    118 ;
0000    119 ;       V02-019 TMK0019         Todd M. Katz            25-Nov-1981
0000    120 ;               Skip any subtrees rooted at null length directory
0000    121 ;               filenames. Such subtrees are pruned, and their
0000    122 ;               contents NOT searched.
0000    123 ;
0000    124 ;       V02-018 TMK0018         Todd M. Katz            18-Nov-1981
0000    125 ;               Initialize certain control fields within the SWB to
0000    126 ;               their correct values for UIC directories even if these
0000    127 ;               fields are not needed for wildcard processing. This is
0000    128 ;               to avoid confusion when these fields are looked at.
0000    129 ;
0000    130 ;       V02-017 TMK0014         Todd M. Katz            12-Nov-1981
0000    131 ;               Change a BBC to a BBCC so that the flag SWB$V_FIRST is
0000    132 ;               always cleared after the first time through RM$NEXTDIR
0000    133 ;
0000    134 ;       V02-016 RAS0040         Ron Schaefer            26-Oct-1981
0000    135 ;               Implement rooted directories for concealed devices.
0000    136 ;               When setting the FID of the MFD, use the saved MFD_FID
0000    137 ;               of the rooted directory.
0000    138 ;
0000    139 ;       V02-015 TMK0010         Todd M. Katz            20_Oct_1981
0000    140 ;
0000    141 ;               A check is made to see whether I/O Rundown is in progress
```

```
0000    142 ;               after every ACP call, and if is, RMOWILD is immediately
0000    143 ;               terminated with a status of RMS$_NMF.
0000    144 ;
0000    145 ;       V02-014 TMK0007         Todd M. Katz            28_Aug_1981
0000    146 ;
0000    147 ;               Performance enhancements. Initially, all leading nonwild
0000    148 ;               tokens are made part of the directory string, and their
0000    149 ;               existance is confirmed. Then, each and every remaining token
0000    150 ;               in the pattern string up to, but not including the first
0000    151 ;               ellipsis, represents the pattern for the directory name that
0000    152 ;               is to match it, and these are resolved. All branches below the
0000    153 ;               level of the root of the first ellipsis must be searched for
0000    154 ;               any and all directories. If there are no ellipsis, or above the
0000    155 ;               root of the first ellipsis, crosswise directory tree traversal
0000    156 ;               can only occur if wild nonellipsis tokens were encountered.
0000    157 ;               After each directory name is resolved, a call is made to a
0000    158 ;               matching routine to determine whether the directory string
0000    159 ;               now matches the entered pattern. This routine returns partial
0000    160 ;               or total success, and it can only return failure when it is
0000    161 ;               impossible to match the trailing tokens while within a bounded
0000    162 ;               ellipsis traversal. NOTE: the number of characters in a
0000    163 ;               directory specification is limited to 81 although theoretically
0000    164 ;               it would be possible to have (8 levels * 9 + 8 ... * 3 + 2)
0000    165 ;               = 98 characters.
0000    166 ;
0000    167 ;       V02-013 TMK0006         Todd M. Katz            21_Aug_1981
0000    168 ;               Complete rewrite. The algorithm upon which this module had
0000    169 ;               been formerly based was scrapped, and a completely new one
0000    170 ;               substituted. The new algorithm guarentees that each and
0000    171 ;               every directory specification matching an inputed pattern
0000    172 ;               is touched only once by performing a "pruned" pre-order
0000    173 ;               traversal, and returning only those directories matching
0000    174 ;               the inputed pattern string. Pruned in the sense that not
0000    175 ;               all directories are touched, but rather, as few as possible.
0000    176 ;               Various optimizations have been implemented to reduce
0000    177 ;               extraneous calls to the ACP, and several more, which are
0000    178 ;               required, will be included within a succeeding enhancement.
0000    179 ;
0000    180 ;               Checked this in as new module. Previous edit history invalid.
0000    181 ;
0000    182 ;       V02-012 KEK0007         K. E. Kinnear           11-Aug-1981
0000    183 ;               Change IFB$L_AS_DEV to IFB$L_PRIM_DEV where necessary.
0000    184 ;
0000    185 ;       V02-011 TMK0005         Todd M. Katz            11_Aug_1981
0000    186 ;               Delete errant ASSUME statement within FIND_DIR. It was
0000    187 ;               no longer needed with the addition of the new field
0000    188 ;               FWA$T_WILD.
0000    189 ;
0000    190 ;       V02-010 TMK0004         Todd M. Katz            10_Aug_1981
0000    191 ;               Correction of the parameter FWA$C_MAXDIRLEN to its
0000    192 ;               correct size of 81 resulted in a much larger SWB.
0000    193 ;               Since only 79 bytes are required ([] are not copied
0000    194 ;               into SWB$T_PATTERN), a change was made to reflect this
0000    195 ;               and to take advantage of a new scratch field, FWA$T_WILD.
0000    196 ;
0000    197 ;       V02-009 TMK0001         Todd M. Katz            29_Jul_1981
0000    198 ;               Fix some problems with handling of bounded elipsis
```

```
0000   199 ;               and multiple wild card directory specifications.
0000   200 ;               Changes included deletion of SWB$B_WCCLEVEL, extension
0000   201 ;               of SWB$B_ROOT to a word, the addition of one flag to
0000   202 ;               SWB$W_FLAGS (SWB$V_ROOT), and minor alterations to
0000   203 ;               the depth-first search algorithm in order to resolve
0000   204 ;               the existing problems. Two routines (UP_PARENT and
0000   205 ;               STRIP_NONWILD) were completely rewritten.
0000   206 ;
0000   207 ;       V02-008 PSK0003         Paulina Knibbe          27-Apr-1981
0000   208 ;               Fix some problems with handling of FWA$B_DIRLN
0000   209 ;
0000   210 ;       V02-007 PSK0002         Paulina Knibbe          18-Nov-1980
0000   211 ;               Clear the wild directory bit for trailing specified
0000   212 ;               directories when there is a bounded elipsis
0000   213 ;
0000   214 ;       V02-006 PSK0001         Paulina Knibbe          12-Nov-1980
0000   215 ;               Set the correct wild directory bit in the FWA if the directory
0000   216 ;               name was originally specified by an elipsis
0000   217 ;
0000   218 ;       V02-005 MCN0002         Maria del C. Nasr       03-Sep-1980
0000   219 ;               Fix branch to prevent infinite loop on parse of
0000   220 ;               [A.B.*]*.* when B does not exist.
0000   221 ;
0000   222 ;       V02-004 REFORMAT        Ron Schaefer    25-Jul-1980
0000   223 ;               Reformat the source
0000   224 ;
0000   225 ;       V003    TMH0003         Tim Halvorsen   05-FEB-1980
0000   226 ;               if sdi device, return with mfd without accessing device.
0000   227 ;
0000   228 ;       V002    TMH0002         Tim Halvorsen   04-JAN-1980
0000   229 ;               fix [*.obj] from failing to do check after copy_nonwild
0000   230 ;               which ensures that the directory up to that point exists.
0000   231 ;               fix [...a...] to return [...a] as well as the subtree
0000   232 ;               below it -- required a check following bounded match
0000   233 ;               to check if the elipsis exit should be taken immediately.
0000   234 ;               add $devdef to generate short literals on checks.
0000   235 ;
0000   236 ;       V001    TMH0001         Tim Halvorsen   12-NOV-1979
0000   237 ;               consolidate multiple asterisks from [*,*] into a single
0000   238 ;               asterisk so that ods-1 can use [*,*] (since character
0000   239 ;               wildcarding is not yet done for ods-1).
0000   240 ;               fix bug in nextdir so that a non-wild directory string
0000   241 ;               is checked immediately to make sure that it exists.
0000   242 ;               fix bug in next_child so that maperr is called with r8
0000   243 ;               pointing to the fab rather than the swb.
0000   244 ;--
```

```
                        .SBTTL  DEFINITIONS

;
; symbol definitions
;

        $FIDDEF
        $DEVDEF                         ; device characteristics
        $FABDEF                         ; fab definitions
        $FIBDEF                         ; fib definitions
        $FSCBDEF                        ; fscb bit definitions
        $FWADEF                         ; fwa definitions
        $IFBDEF                         ; ifab definitions
        $IMPDEF                         ; i/o segment definitions
        $IODEF                          ; i/o function codes
        $NAMDEF                         ; nam definitions
        $RMSDEF                         ; rms error codes
        $SSDEF                          ; system error codes
        $SWBDEF                         ; swb definitions
```

```
                                0000   266          .SBTTL  RM$INIT_SWB, Initialize SWB for Wildcarding
                                0000   267
                                0000   268 ;++
                                0000   269 ;
                                0000   270 ; RM$INIT_SWB: initialize swb for wildcarding
                                0000   271 ;
                                0000   272 ;       this routine takes the parsed directory specification,
                                0000   273 ;       and recreates the input file directory specification, minus
                                0000   274 ;       the delimiter brackets, in the SWB for later use. It also
                                0000   275 ;       initializes various fields within the SWB for use later on.
                                0000   276 ;
                                0000   277 ; Inputs:
                                0000   278 ;
                                0000   279 ;       r9  = ifab address
                                0000   280 ;       r10 = fwa address
                                0000   281 ;
                                0000   282 ; Outputs:
                                0000   283 ;       R0 success/fail
                                0000   284 ;--
                                0000   285
                                0000   286 RM$INIT_SWB::
                       58   DD  0000   287          PUSHL   R8                      ; save fab
                                0002   288
                                0002   289 ;
                                0002   290 ; Allocate swb buffer for expanded string
                                0002   291 ;
                                0002   292
                          58 4C AA   D0  0002   293          MOVL    FWA$L_SWB_PTR(R10),R8   ; already have one?
                                19   12  0006   294          BNEQ    5$
                             52 0052 8F   3C  0008   295          MOVZWL  #SWB$C_BLN/4,R2         ; size of structure
                              00000000'EF   16  000D   296          JSB     RM$GETBLK1              ; get space
                                    7D   50  E9  0013   297          BLBC    R0,90$                  ; quit if no room
                                08 A1    2A  90  0016   298          MOVB    #SWB$C_BID,SWB$B_BID(R1); set block id
                                4C AA    51  D0  001A   299          MOVL    R1,FWA$L_SWB_PTR(R10)   ; set ptr
                                   58    51  D0  001E   300          MOVL    R1,R8                   ; mov ptr to right register
                                         0021   301
                                         0021   302 ;
                                         0021   303 ; If the directory specification was not in UIC format, then recreate it in
                                         0021   304 ; the SWB keeping track of the number of nonelllipses tokens, the token number
                                         0021   305 ; of the first ellipsis (if there is one), and setting various flags as
                                         0021   306 ; required.
                                         0021   307 ;
                                         0021   308
                                      68 D4  0021   309 5$:       CLRL    (R8)                    ; clear flags
                                   04 A8 D4  0023   310          CLRL    SWB$B_MINIMUM(R8)       ; and counters
                                   05 AA 90  0026   311          MOVB    FWA$B_DIRWCFLGS(R10),-  ; save the wild-card context
                                   07 A8     0029   312                  SWB$B_DIRWCFLGS(R8)     ; present on entry in the SWB
                                53 48 A8 9E  002B   313          MOVAB   SWB$T_PATTERN_BUF(R8),R3; initialize pattern descriptor with
                                10 A8 53 D0  002F   314          MOVL    R3,SWB$Q_PATTERN+4(R8)  ; address of pattern buffer
                             56 0130 CA 9E  0033   315          MOVAB   FWA$Q_DIR1(R10),R6      ; obtain address of first descriptor
                                5B 6A 1B E0  0038   316          BBS     #FWA$V_GRPMBR,(R10),70$ ; branch if UIC format
                                   57 08 D0  003C   317          MOVL    #FWA$C_MAXSUBDIR+1,R7   ; maximum number of dirs in a spec
                                         003F   318
                                      66 B5  003F   319 10$:      TSTW    (R6)                    ; end of directory names?
                                      26 13  0041   320          BEQL    40$                     ; branch if so
                                   03 A8 96  0043   321          INCB    SWB$B_TOKENS_LEFT(R8)   ; increment number of tokens counter
                          63 04 B6    66 28  0046   322          MOVC3   (R6),@4(R6),(R3)        ; copy current directory into buffer
```

```
                83   2E   90   004B   323          MOVB    #^A/./,(R3)+                    ; and append delimiter
             0F 66   10   E5   004E   324          BBCC    #FSCB$V_ELIPS,(R6),30$          ; ellipsis following this directory?
                                0052   325                                                  ;  (clear it so it's not displayed again)
                      06   E2   0052   326          BBSS    #SWB$V_ELLIPSIS_EXISTS,-        ; if this is the very first ellipsis
                   06 68        0054   327          (R8),20$                                ; encountered, set the ellipsis exists
                   03 A8   81   0056   328          ADDB3   SWB$B_TOKENS_LEFT(R8),-         ; flag, and save the token number of
                06 A8      01        0059   329          #1,SWB$B_FIRST_E(R8)            ; ellipsis for future reference
             83    2E2E   8F   B0   005C   330  20$:    MOVW    #^A/../,(R3)+                   ; make delimiter into ellipsis
                      66   B4   0061   331  30$:    CLRW    (R6)                            ; initialize this slot to null
                   56   08   C0   0063   332          ADDL2   #8,R6                           ; skip to next descriptor
                      D6   57   F5   0066   333          SOBGTR  R7,10$                          ; loop until done
                                0069   334
          FE A3      2E2E   8F   B1   0069   335  40$:    CMPW    #^A/../,-2(R3)                  ; is there a trailing ellipsis?
                      02   13   006F   336          BEQL    50$                             ; yes leave it
                      53   D7   0071   337          DECL    R3                              ; no, remove trailing delimiter
                                0073   338
                   08   57   83   0073   339  50$:    SUBB3   R7,#FWA$C_MAXSUBDIR+1,-         ; compute the maximum success level
                05 A8           0076   340          SWB$B_MAXIMUM(R8)                       ; presuppossing there are no ellipses
                      06   E1   0078   341          BBC     #SWB$V_ELLIPSIS_EXISTS,-        ; but if an ellipsis was encountered,
                   04 68        007A   342          (R8),60$                                ; then the maximum level is reset to
                   05 A8   90   007C   343          MOVB    #FWA$C_MAXSUBDIR+1,-            ; allow for searches to the lowest
                05 A8           007E   344          SWB$B_MAXIMUM(R8)                       ; subdirectory level supported
                                0080   345
       0C A8   53   10 AB   C3   0080   346  60$:    SUBL3   SWB$Q_PATTERN+4(R8),R3,-        ; compute the length of the pattern
                                0086   347          SWB$Q_PATTERN(R8)                       ; and put it in the pattern descriptor
                                0086   348          SSB     #SWB$V_FIRST,(R8)               ; set first time through bit
                    0361   30   008A   349          BSBW    PARSE_PATTERN                   ; parse the very first pattern
                   2E AA   94   008D   350          CLRB    FWA$B_DIRLEN(R10)               ; mark no directory names yet
                                0090   351          RMSSUC
                   58 8ED0   0093   352  90$:    POPL    R8                              ; restore fab
                         05   0096   353          RSB
```

```
                        0097   355
                        0097   356 ;
                        0097   357 ; if the directory specification is in UIC format, then expand both group
                        0097   358 ; and member strings to three characters each, and store them together as
                        0097   359 ; the only token, provided neither is wild. if either or both is wild, then
                        0097   360 ; the wild strings are copied directly into the buffer without expansion,
                        0097   361 ; and one token is created as before.
                        0097   362 ;
                        0097   363
          57  28  D0     0097   364 70$:    MOVL    #FWA$V_WILD_GRP,R7      ; start of directory wildcard bits
              16  10     009A   365         BSBB    UIC_EXPAND              ; expand group portion
              57  D6     009C   366         INCL    R7                      ; skip to next wildcard bit
              12  10     009E   367         BSBB    UIC_EXPAND              ; expand member portion
        0202  8F  B0     00A0   368         MOVW    #^X0202,-               ; both the minimum and maximum success
          04  A8         00A4   369                 SWB$B_MINIMUM(R8)       ; levels are 2 for UIC directories
2A2A 8F   FE  A3  B1     00A6   370         CMPW    -2(R3),#^A'**'          ; last 2 characters = '*'?
              D2  12     00AC   371         BNEQ    60$                     ; branch if not
              53  D7     00AE   372         DECL    R3                      ; if so, 1 asterisk will do just fine
              CE  11     00B0   373         BRB     60$                     ; setup pattern descriptor
                        00B2   374
                        00B2   375 ;
                        00B2   376 ; expand the nonwild group/member string out to three characters, and place
                        00B2   377 ; it in the buffer. if the group/member is wild, copy it into the pattern
                        00B2   378 ; buffer as is.  NOTE: Don't use SOBGEQ since the descriptor will have flags
                        00B2   379 ;
                        00B2   380
                        00B2   381 UIC_EXPAND:
          50  86  D0     00B2   382         MOVL    (R6)+,R0                ; get length of string
        0D 6A  57  E0     00B5   383         BBS     R7,(R10),30$            ; skip leading zeros if wild
      51  03  50  A3     00B9   384         SUBW3   R0,#3,R1                ; number of leading zeros to insert
              05  11     00BD   385         BRB     20$                     ; get into loop
          83  30  90     00BF   386 10$:    MOVB    #^A/0/,(R3)+            ; store leading zero
              51  B7     00C2   387         DECW    R1                      ; count down
              F9  14     00C4   388 20$:    BGTR    10$                     ; as many as needed
  63      96  50  28     00C6   389 30$:    MOVC3   R0,@(R6)+,(R3)          ; store rest of string
                  05     00CA   390         RSB
```

```
                     00CB    392          .SBTTL  RMSNEXTDIR, Get Next Directory to Search
                     00CB    393
                     00CB    394 ;++
                     00CB    395 ;
                     00CB    396 ; RMSNEXTDIR: Get next directory to search
                     00CB    397 ;
                     00CB    398 ;       get the next directory which matches the wildcard
                     00CB    399 ;       pattern in the directory specification.
                     00CB    400 ;
                     00CB    401 ; inputs:
                     00CB    402 ;
                     00CB    403 ;       R9  = ifab address
                     00CB    404 ;       R10 = fwa address
                     00CB    405 ;       R11 = impure area address
                     00CB    406 ;
                     00CB    407 ; outputs:
                     00CB    408 ;
                     00CB    409 ;       R8 - destroyed
                     00CB    410 ;       R0 = nmf if no more files to search, else status
                     00CB    411 ;
                     00CB    412 ;       the fib is updated with the did of the next
                     00CB    413 ;       directory to be used in file searches.
                     00CB    414 ;
                     00CB    415 ;--
                     00CB    416
                     00CB    417 RMSNEXTDIR::
          58   4C AA   D0  00CB    418          MOVL    FWA$L_SWB_PTR(R10),R8   ; get swb address
                 0B   13  00CF    419          BEQL    5$                      ; branch if isn't one
                     00D1    420
                     00D1    421 ;
                     00D1    422 ; the very first time this routine is called, all leading nonwild tokens are
                     00D1    423 ; copied into FWA descriptors, their existance is verified, and a starting
                     00D1    424 ; DID is setup in the FIB for the remainder of the token string. at this
                     00D1    425 ; point, the minimum directory traversal string can be set, and a check is
                     00D1    426 ; made as to whether the current directory specification is sufficient to
                     00D1    427 ; match the entered pattern. note that if the base ends up as the MFD, the
                     00D1    428 ; initial match attempt can be skipped (there is nothing to match) and
                     00D1    429 ; the first UFD can be retrieved. if all tokens are nonwild, then there is
                     00D1    430 ; nothing more to do and a status of success can be returned.
                     00D1    431 ;
                     00D1    432
        2E 68   05   E5  00D1    433          BBCC    #SWB$V_FIRST,(R8),20$   ; do only if first time through
              0421   30  00D5    434          BSBW    SET_BASE                ; copy all leading nonwild tokens,
           07 50     E8  00D8    435          BLBS    R0,10$                  ; find the base DID, and return
                     05  00DB    436          RSB                             ; if thereare any problems
                     00DC    437
                     00DC    438 5$:      RMSERR  NOVALPRS                ; no preceding $parse
                     05  00E1    439          RSB
                     00E2    440
          2E AA      90  00E2    441 10$:     MOVB    FWA$B_DIRLEN(R10),-     ; set the minimum success level
          04 A8          00E5    442                  SWB$B_MINIMUM(R8)       ; for traversal
          01 A8      95  00E7    443          TSTB    SWB$B_PATLEN(R8)        ; if all tokens in the entered pattern
             06      13  00EA    444          BEQL    11$                     ;  were nonwild then immediately return
          2E AA      91  00EC    445          CMPB    FWA$B_DIRLEN(R10),-     ; if we are at the bottom also return
             08          00EF    446                  #FWA$C_MAXSUBDIR+1
             03      12  00F0    447          BNEQ    12$                     ; else continue
           00FB      31  00F2    448 11$:     BRW     110$                    ; return success and the DID
```

```
                     00F5   449
                     00F5   450 12$:    TSTB    FWA$B_DIRLEN(R10)          ; if there were leading nonwild tokens
          2E AA   95 00F5   450          BEQL    15$                        ; with other tokens following them
            03   13 00F8   451          BRW     70$                        ; then check for pattern-string match
          0099   31 00FA   452 15$:     MOVL    (R8),SWB$L_SCRATCH_PAT(R8) ; else move pattern of first dir name
    14 A8   68   D0 00FD   453          BRB     40$                        ; to search for into field & go search
            2E   11 0101   454
                     0103   455
                     0103   456 ;
                     0103   457 ; traverse down one level by getting the first subdirectory of the current
                     0103   458 ; directory and appending it. under the following conditions, downward
                     0103   459 ; traversal is aborted in favor of traversing across: the SWB traversal flag
                     0103   460 ; is set; the current directory's level is the maximum traversal level; the
                     0103   461 ; current directory has no subdirectories; the current directory represents
                     0103   462 ; the MFD (i.e. the sole directory is the current directory 000000.dir;1).
                     0103   463 ; if the token string has been exhausted, or if the current directory has no
                     0103   464 ; subdirectories and is at the minimum traversal level, then there are no
                     0103   465 ; directories to be located.
                     0103   466 ;
                     0103   467
          01 A8   95 0103   468 20$:    TSTB    SWB$B_PATLEN(R8)           ; if token string has been exhausted
            03   12 0106   469          BNEQ    25$                        ; then signal no more files,
          00D5   31 0108   470          BRW     90$                        ; otherwise continue
    2F 68   04   E4 010B   471 25$:     BBSC    #SWB$V_TRAVERSE,(R8),45$   ; if we are to traverse, then traverse
          05 A8   91 010F   472          CMPB    SWB$B_MAXIMUM(R8),-        ; if we have reached the maximum
          2E AA      0112   473                  FWA$B_DIRLEN(R10)          ; traversal level then branch so as to
            28   13 0114   474          BEQL    45$                        ; begin traversing across
                     0116   475          SSB     #SWB$V_ELLIPSIS,-          ; only traverse down other than first
                     0116   476                  SWB$L_SCRATCH_PAT(R8)      ; time through if have seen an ellipsis
                     011B   477
                     011B   478 ;
                     011B   479 ; if the base directory (i.e. the directory whose file ID is in the DID
                     011B   480 ; of the FIB) is not the MFD, then continue the downward traversal
                     011B   481 ;
                     011B   482
    54    0130 CA   7D 011B   483 30$:    MOVQ    FWA$Q_DIR1(R10),R4         ; get first directory's descriptor
            0298   30 0120   484          BSBW    CHK_MFD                    ; check for mfd,  if so
              0C   12 0123   485          BNEQ    40$                        ; then continue the downward traversal
          04 A8   97 0125   486          DECB    SWB$B_MINIMUM(R8)          ; otherwise decrement the minimum
              06   E0 0128   487          BBS     #SWB$V_ELLIPSIS_EXISTS,-   ; traversal level, and the maximum
          20 68      012A   488                  (R8),55$                   ; traversal level if there are any
          05 A8   97 012C   489          DECB    SWB$B_MAXIMUM(R8)          ; ellipsis in the entered pattern and
              1B   11 012F   490          BRB     55$                        ; begin crosswise traversal
                     0131   491
        0204 CA   D4 0131   492 40$:    CLRL    FWA$T_FIBBUF+-             ; clear the wildcard context within the
                     0135   493                  FIB$L_WCC(R10)             ; FIB to get the first subdirectory
            00BC   30 0135   494          BSBW    NEXT_SUBDIR                ; get first subdirectory of current dir
          53 50   E9 0138   495          BLBC    R0,67$                     ; branch if error occurred otherwise
          58 51   E8 013B   496          BLBS    R1,70$                     ; see if dir spec now matches pattern
```

```
                     013E   498
                     013E   499 ;
                     013E   500 ; traverse across the directory tree. if we are now at the minimum traversal
                     013E   501 ; level, then there are no more directories to find, otherwise, we back up to
                     013E   502 ; the level of a wild directory by stripping off the lowest level directory,
                     013E   503 ; calling match to retrieve the characteristics of the directory name to be
                     013E   504 ; searched for, and repeating this process as long as the pattern returned
                     013E   505 ; is nonwild. once the level of a wild directory has been reached, the next
                     013E   506 ; directory at the same level matching the returned pattern is found by using
                     013E   507 ; the stripped off directory name to resynchronize the ACP. note that if any
                     013E   508 ; time match returns a status of total success, this can only be do to the
                     013E   509 ; presence of an ellipsis, and any directory will satisfy the pattern.
                     013E   510 ;
                     013E   511
                     013E   512 45$:    CSB     #SWB$V_VALID_DID,(R8)     ; DID is no longer valid
          2E AA   91 0142   513 50$:    CMPB    FWA$B_DIRLEN(R10),-       ; if at minimum traversal level and
          04 A8      0145   514                    SWB$B_MINIMUM(R8)       ; no subdirectory was found then exit
             03   14 0147   515          BGTR    55$                       ; else continue with the crosswise
           0094   31 0149   516          BRW     90$                       ; traversal across the directory tree
                     014C   517
           0302   30 014C   518 55$:    BSBW    PREV_DIR                  ; strip off the lowest level directory
        7E   54   7D 014F   519          MOVQ    R4,-(SP)                  ; and save its descriptor
             68   DD 0152   520          PUSHL   (R8)                      ; save current token context
     56  04 A8   9A 0154   521          MOVZBL  SWB$B_MINIMUM(R8),R6      ; set descriptor index of 1st wild dir
           0184   30 0158   522          BSBW    MATCH                     ; retrieve pattern of dir to search for
             68 8ED0 015B   523          POPL    (R8)                      ; restore current token context and
        54   8E   7D 015E   524          MOVQ    (SP)+,R4                  ; descriptor of stripped of directory
             07 51   E8 0161   525          BLBS    R1,60$                    ; if total success returned then branch
             02   E1 0164   526          BBC     #SWB$V_WILD,-             ; if the pattern returned was of a
        E3 14 A8      0166   527                    SWB$L_SCRATCH_PAT(R8),55$       ; nonwild token then repreat process
             05   11 0169   528          BRB     65$                       ; otherwise continue across traversal
                     016B   529 60$:    SSB     #SWB$V_ELLIPSIS,-         ; when total success is returned, set
                     016B   530                    SWB$L_SCRATCH_PAT(R8)     ; it up so anything will match pattern
                     0170   531
        6C A9   54   06   C1 0170   532 65$:    ADDL3   #6,R4,IFB$L_RNS_LEN(R9) ; the stripped directory's length (plus
                65   54   28 0175   533          MOVC3   R4,(R5),-         ; 6 for .dir;1) & its name are together
                   04B6   CA 0178   534                  FWA$T_NAMEBUF(R10)      ; used to resynchronize the ACP
     63 20203138 5249442E   8F 7D 017B   535          MOVQ    #^A\.DIR;1  \,(R3)      ; append .dir;1 to the directory name
                 0204   CA 01 D0 0186   536          MOVL    #1,FWA$T_FIBBUF+-       ; indicate that the ACP is to
                                 018B   537                  FIB$L_WCC(R10)          ; resynchronize by setting a bit
                        0066   30 018B   538          BSBW    NEXT_SUBDIR             ; get next directory at the same level
                     2C 50   E9 018E   539 67$:    BLBC    R0,80$                  ; error   - go handle it
                     02 51   E8 0191   540          BLBS    R1,70$                  ; success - check for pattern match
                        A8   11 0194   541          BRB     45$                     ; failure - continue 1 level up
```

```
                        0196   543
                        0196   544 ;
                        0196   545 ; determine the match status of the current directory specification against
                        0196   546 ; the pattern. there are three possible outcomes. the match could be totally
                        0196   547 ; successful, in which case the next directory meeting the stated requirements
                        0196   548 ; has been found. the match could be partially successful indicating that
                        0196   549 ; while the current specification matches the leading part of the pattern
                        0196   550 ; it is insufficient for a total match, and downward traversing should
                        0196   551 ; continue. finally, the match could be a total failure. this can only occur
                        0196   552 ; if we are traversing within a bounded ellipsis, and can never hope to
                        0196   553 ; match the trailing tokens from the current level. we continue by traversing
                        0196   554 ; across.
                        0196   555 ;
                        0196   556
4C 6A    1B    E0       0196   557 70$:    BBS     #FWA$V_GRPMBR,(R10),100$; if UIC then total success
         68    DD       019A   558         PUSHL   (R8)                    ; save the current token information
56   04  A8    9A       019C   559         MOVZBL  SWB$B_MINIMUM(R8),R6    ; set descriptor index of 1st wild dir
     07  A8    90       01A0   560         MOVB    SWB$B_DIRWCFLGS(R8),-   ; reinitialize the directory wildcard
     05  AA             01A3   561                 FWA$B_DIRWCFLGS(R10)    ; context to its value before 1st srch
       0137    30       01A5   562         BSBW    MATCH                   ; determine state of the match
         68 8ED0       01A8   563         POPL    (R8)                    ; restore current token information
     38  51    E8       01AB   564         BLBS    R1,100$                 ; total success   - find dir's DID
     07  A8    90       01AE   565         MOVB    SWB$B_DIRWCFLGS(R8),-   ;                   restore wild card
     05  AA             01B1   566                 FWA$B_DIRWCFLGS(R10)    ;                   directory context
     96  50    E9       01B3   567         BLBC    R0,55$                  ; failure         - traverse across
                        01B6   568         CSB     #SWB$V_VALID_DID,(R8)   ; partial success - set DID invalid
       FF5E    31       01BA   569         BRW     30$                     ;                   and traverse down
```

```
                     01BD    571
                     01BD    572 ;
                     01BD    573 ; if an error occurred during the attempt to find the next directory, or the
                     01BD    574 ; DID of the next directory, traversal of the current subtree must be aborted
                     01BD    575 ; to avoid looping on the same error. if the directory specification on entry
                     01BD    576 ; was wild and the directory was not found, or was do to illegal directory
                     01BD    577 ; format, this traversal takes place immediately. in all other circumstances
                     01BD    578 ; the error is immediately reported.
                     01BD    579 ;
                     01BD    580
                     01BD    581 80$:     SSB     #SWB$V_TRAVERSE,(R8)        ; set the SWB traverse flag
          1C   E1    01C1    582          BBC     #FWA$V_WILD_DIR,-           ; if the directory specification was
      2E 6A            01C3    583                  (R10),T20$                 ; nonwild then report the error
   51 24 A9    D0      01C5    584          MOVL    IFB$L_LAST_FAB(R9),R1       ; obtain the address of the FAB
      0C A1    B1      01C9    585          CMPW    FAB$L_STV(R1),-             ; if the directory or DID could not
    0910 8F            01CC    586                  #SS$_NOSUCHFILE            ; be found then continue the
         08    13      01CF    587          BEQL    85$                         ; crosswise traversal one level up
      0C A1    B1      01D1    588          CMPW    FAB$L_STV(R1),-             ; if the error was not do to illegal
    0828 8F            01D4    589                  #SS$_BADIRECTORY           ; directory format then report the
         1A    12      01D7    590          BNEQ    120$                        ; error immediately
                     01D9    591 85$:     CSB     #SWB$V_TRAVERSE,(R8)        ; clear traversal bit,
       FF5E    31      01DD    592          BRW     45$                         ; and go traverse
                     01E0    593
                     01E0    594 ;
                     01E0    595 ; if there are no more files to search, return indicating no more files.
                     01E0    596 ;
                     01E0    597
                     01E0    598 90$:     RMSERR  NMF                         ; indicate that there are no more
                05   01E5    599          RSB                                 ; directories to search and return
                     01E6    600
                     01E6    601 ;
                     01E6    602 ; if a directory specification has been found that matches the entered pattern,
                     01E6    603 ; find the DID of the directory specification (unless it is still valid),
                     01E6    604 ; and return success.
                     01E6    605 ;
                     01E6    606
06 68    05   E4    01E6    607 100$:    BBSC    #SWB$V_FIRST,(R8),110$      ; first time through then DID is valid
        0326   30    01EA    608          BSBW    FIND_DIR                    ; find the DID of the directory spec
      CD 50    E9    01ED    609          BLBC    R0,80$                      ; go handle any errors encountered
                     01F0    610 110$:    RMSSUC                              ; else indicate success and
                05   01F3    611 120$:    RSB                                 ; return
                     01F4    612
```

```
                                        .SBTTL  NEXT_SUBDIR, Find the Next Subdirectory

                                ;++
                                ;
                                ; NEXT_SUBDIR:  find the next subdirectory
                                ;
                                ;       find the next subdirectory of the current directory specification.
                                ;       the next subdirectory might be the first subdirectory within the
                                ;       current specification, if we are currently traversing down the
                                ;       directory tree, or it might be the next subdirectory after the last
                                ;       subdirectory located within this directory specification, if we are
                                ;       traversing across the directory tree. if a subdirectory is found,
                                ;       it is appended to the current directory specification.
                                ;
                                ; inputs:
                                ;
                                ;       r8  = SWB address
                                ;       r9  = IFAB address
                                ;       r10 = FWA address
                                ;
                                ; outputs:
                                ;
                                ;       r0 = status
                                ;       r1 = false if no subdirectories left
                                ;            true if subdirectory located
                                ;
                                ;--
                                NEXT_SUBDIR:
                                ;
                                ; construct the file name to search for within the current directory
                                ; specification. the SWB field SWB$L_SCRATCH_PAT contains information about
                                ; the subdirectory we are to search for. *.dir;1 is always searched for
                                ; whenever we are "within" an ellipsis. or an ellipsis has been encountered
                                ; in the building of the current specification; otherwise pattern information
                                ; is present within within the FWA field.
                                ;

      53    18 A8    9E        MOVAB   SWB$T_SCRATCH_BUF(R8),R3; obtain the address of the buffer
  018C CA     53    D0        MOVL    R3,FWA$Q_RNS+4(R10)     ; and move it into the descriptor
              00    E1        BBC     #SWB$V_ELLIPSIS,-       ; if the pattern in the buffer is
          05 14 A8              SWB$L_SCRATCH_PAT(R8),5$        ; that of an ellipsis, then we are
           83    2A    90      MOVB    #^A'*',(R3)+            ; to search for *.dir;1 so we move a *
                 0E    11      BRB     10$                     ; into the buffer and go append .dir;1

      50    15 A8    9A        MOVZBL  SWB$L_SCRATCH_PAT+1(R8),R0      ; create a descriptor of the directo
      51    16 A8    9A        MOVZBL  SWB$L_SCRATCH_PAT+2(R8),R1      ; we are to search for in r0/r1 and
63    10 B841    50    28      MOVC3   R0,@SWB$Q_PATTERN+4-    ; move the name of the directory
                                (R8)[R1],(R3)           ; into the buffer

 83   5249442E 8F    D0        MOVL    #^A\.DIR\,(R3)+         ; append the file type to the dir name
      83    313B 8F  B0        MOVW    #^A';1',(R3)+           ; append the version number to the name
0188 CA     53  018C CA  C3    SUBL3   FWA$Q_RNS+4(R10),R3,-   ; compute the length of the dir name to
                                FWA$Q_RNS(R10)          ; be searched for into its descriptor

                                ;
```

```
                     0229   671 ; the DID of the current directory specification must be found and placed
                     0229   672 ; in the FIB in order to perform searches using the ACP.
                     0229   673 ;
                     0229   674
0A 68    05   E4     0229   675          BBSC    #SWB$V_FIRST,(R8),15$   ; if this is the first time through
           07   E0     022D   676          BBS     #SWB$V_VALID_DID,-      ; or the DID is still valid, then it
         06 68          022F   677                  (R8),15$                ; is not necessary to find the DID
          02DF   30     0231   678          BSBW    FIND_DIR                ; if there is any problem finding the
        7A 50    E9     0234   679          BLBC    R0,70$                  ; DID of the current specification
                        0237   680                                          ; return immediately, else continue
                        0237   681
                        0237   682 ;
                        0237   683 ; required as input to the ACP (or the routine which is responsible for
                        0237   684 ; calling the ACP) is that a descriptor of the FIB exist in FWA$Q_FIB(R10),
                        0237   685 ; that name string wildcarding be turned on within the FIB, and that the
                        0237   686 ; descriptor of the result buffer be initialized with the size of the
                        0237   687 ; maximum possible file name string. RM$STALL also requires that general
                        0237   688 ; register R8 contains the address of the FAB.
                        0237   689 ;
                        0237   690
          40 8F    9A     0237   691 15$:     MOVZBL  #FIB$C_LENGTH,-         ; load the length of the FIB
          10 AA           023A   692                  FWA$Q_FIB(R10)          ; into the FIB descriptor
        01F4 CA    9E     023C   693          MOVAB   FWA$T_FIBBUF(R10),-     ; load the address of the FIB
          14 AA           0240   694                  FWA$Q_FIB+4(R10)        ; into the FIB descriptor
                     3C     0242   695          MOVZWL  #FWA$S_NAMEBUF+-        ; initialize the length field
                        0243   696                  FWA$S_TYPEBUF+-         ;  of the result descriptor
                        0243   697                  FWA$S_VERBUF,-          ;  with the maximum size that
    0170 CA   012E 8F   0243   698                  FWA$Q_NAME(R10)         ;  a file name can be
              0100 8F   B0     0249   699          MOVW    #FIB$M_WILD,-           ; set the bit within the FIB which
              0208 CA          024D   700                  FWA$T_FIBBUF+-          ; indicates that a wildcard name
                        0250   701                  FIB$W_NMCTL(R10)        ; search is to be performed
                        0250   702
```

```
                     0250   704 ;
                     0250   705 ; the ACP will search for the next directory within the current directory
                     0250   706 ; specification, whose DID is already resident within the FIB, because the
                     0250   707 ; name string wildcard bit has been set and the input descriptor points to a
                     0250   708 ; buffer containing the string to be searched for. if the FIB field FIB$L_WCC
                     0250   709 ; has been cleared, then to the ACP the next directory is the very first
                     0250   710 ; subdirectory in the current directory specification which matches the pattern
                     0250   711 ; in the buffer. on the other hand, if this same field has been set to one,
                     0250   712 ; then the next directory is the first subdirectory in the current directory
                     0250   713 ; specification which matches the pattern in the buffer, and follows the
                     0250   714 ; subdirectory described by the descriptor FWA$Q_NAME(R10) (contains the
                     0250   715 ; address of the buffer containing the subdirectory's name), and the longword
                     0250   716 ; IFB_L_RNS_LEN(R9) (contains the length of the subdirectory's name). the
                     0250   717 ; following are the parameters required by the ACP to be present on the
                     0250   718 ; current stack when it is called.
                     0250   719 ;
                     0250   720 ;       P1      #IO$_ACCESS             - ACP function code
                     0250   721 ;       P2      FWA$Q_RNS(R10)          - address of the input name descriptor
                     0250   722 ;       P3      IFB$L_RNS_LEN(R9)       - longword to recieve result name's
                     0250   723 ;                                         length, and input as previous
                     0250   724 ;                                         position in directory
                     0250   725 ;       P4      FWA$Q_NAME(R10)         - address of result descriptor and
                     0250   726 ;                                         input to ACP as previous position
                     0250   727 ;                                         in directory
                     0250   728 ;       P5      0
                     0250   729 ;       P6      0
                     0250   730 ;
                     0250   731
     58   24 A9   D0 0250   732 20$:    MOVL    IFB$L_LAST_FAB(R9),R8   ; of the FAB into R8, and go
          7E   7C 0254   733             CLRQ    -(SP)                   ; place P5 & P6 on stack
        0170 CA   9F 0256   734             PUSHAB  FWA$Q_NAME(R10)         ; push result descriptor address - P4
          6C A9   9F 025A   735             PUSHAB  IFB$L_RNS_LEN(R9)       ; push length field - P3
        0188 CA   9F 025D   736             PUSHAB  FWA$Q_RNS(R10)          ; push input name descriptor - P2
          50    32 3C 0261   737             MOVZWL  #IO$_ACCESS,R0          ; set ACP function code in R0
00000000'EF   16 0264   738             JSB     RM$FCPFNC               ; call the ACP and wait for reply
                     026A   739
          04   E0 026A   740             BBS     #IMP$V_IORUNDOWN,-      ; if I/O Rundown is in progress,
        44 6B           026C   741             (R11),75$               ; then RMOWILD is prematurely aborted
     58   4C AA   D0 026E   742             MOVL    FWA$L_SWB_PTR(R10),R8   ; recover SWB address
          43 50   E9 0272   743             BLBC    R0,80$                  ; branch if no errors
                     0275   744
                     0275   745 ;
                     0275   746 ; the ACP found the next subdirectory - extract it for processing. if the
                     0275   747 ; directory name has a null length, go resynchronize the ACP and find the
                     0275   748 ; next directory. subtrees rooted at null length directory filenames are
                     0275   749 ; pruned.
                     0275   750 ;
                     0275   751
     54   6C A9   3C 0275   752 40$:    MOVZWL  IFB$L_RNS_LEN(R9),R4    ; get a descriptor of the result's file
  55   04B6 CA   9E 0279   753             MOVAB   FWA$T_NAMEBUF(R10),R5   ; name by chopping off the substring
          54    06 C2 027E   754             SUBL2   #6,R4                   ; .dir;1 and resetting the length
          1E   13 0281   755             BEQL    55$                     ; if null length, go resynch ACP
                     0283   756
                     0283   757 ;
                     0283   758 ; if the file found by the ACP was the MFD itself (i.e. the file 000000.dir;1)
                     0283   759 ; then skip it by recalling the ACP giving it this file as its current position
                     0283   760 ; within the MFD. this is so directory specifications such as [*...], [*], and
```

```
                     0283   761 ; [*,*] won't return the MFD.
                     0283   762 ;
                     0283   763
                0135 30   0283   764          BSBW    CHK_MFD                 ; check for mfd, if so the reset
                  19 13   0286   765          BEQL    55$                     ; and resynchronize the ACP
                     0288   766
                     0288   767 ;
                     0288   768 ; if the directory was given in UIC format (but wild !!), then make sure
                     0288   769 ; only numeric directory names will be accepted as valid UIC format names.
                     0288   770 ;
                     0288   771
              1C 6A  1B E1   0288   772 45$:    BBC     #FWA$V_GRPMBR,(R10),60$ ; branch if not in UIC format
                  50 54 7D   028C   773          MOVQ    R4,R0                   ; duplicate result name descriptor
                     028F   774
                  30 61 91   028F   775 50$:    CMPB    (R1),#^A'0'             ; if the name coallates lower then any
                     0D 1F   0292   776          BLSSU   55$                     ; valid UIC name - traverse across
                  37 81 91   0294   777          CMPB    (R1)+,#^A'7'            ; if the result name coallates higher
                     42 1A   0297   778          BGTRU   90$                     ; then any valid UIC name, then
                   F3 50 F5  0299   779          SOBGTR  R0,50$                  ; go return no more files
                     029C   780
                  06 54 D1   029C   781          CMPL    R4,#6                   ; if the result name is 6 numbers long
                     07 13   029F   782          BEQL    60$                     ; then it is a valid UIC name
                     02A1   783
            0204 CA 01 D0    02A1   784 55$:    MOVL    #1,FWA$T_FIBBUF+-       ; resynchronize the ACP to continue
                     02A6   785                  FIB$L_WCC(R10)          ; the search, reload the address
                     A8 11   02A6   786          BRB     20$                     ; look for the next subdirectory
                     02A8   787
                     02A8   788 ;
                     02A8   789 ; append the new subdirectory name to the current directory specification
                     02A8   790 ; and return success - i.e. a subdirectory was found.
                     02A8   791 ;
                     02A8   792
                0223 30   02A8   793 60$:    BSBW    APPEND_DIR              ; append the subdirectory
              51   01 D0   02AB   794          MOVL    #1,R1                   ; indicate that a subdirectory was
              50   01 D0   02AE   795 65$:    MOVL    #1,R0                   ; found by setting both r0 and r1
                     05   02B1   796 70$:    RSB                             ; before returning
                     02B2   797
                     02B2   798 75$:    RMSERR  NMF                     ; setting a default error of no more
                     05   02B7   799          RSB                             ; files in R0 and returning
                     02B8   800
                     02B8   801 ;
                     02B8   802 ; we divide ACP errors into two catagories - fatal and nonfatal. nonfatal
                     02B8   803 ; errors, such as no more subdirectories within the current directory
                     02B8   804 ; specification, are handled by this module. fatal errors maybe handled by this
                     02B8   805 ; module depending upon their nature. both types of errors usaully result in
                     02B8   806 ; the traversal continuing, but in a crosswise manner one level up from the
                     02B8   807 ; level of the current directory specification.
                     02B8   808 ;
                     02B8   809
          0910 8F    50 B1   02B8   810 80$:    CMPW    R0,#SS$_NOSUCHFILE      ; if the ACP was unable to find a
                     1C 13   02BD   811          BEQL    90$                     ; subdirectory return no such file
          0930 8F    50 B1   02BF   812          CMPW    R0,#SS$_NOMOREFILES     ; if the ACP found no more files in the
                     15 13   02C4   813          BEQL    90$                     ; sequence return no more files
                     58 DD   02C6   814          PUSHL   R8                      ; save the SWB address
                58   24 A9   02C8   815          MOVL    IFB$L_LAST_FAB(R9),R8   ; storing other errors requires a(FAB)
                     02CC   816          RMSERR  FND,RT                  ; set default error
           00000000'EF 16   02D1   817          JSB     RMS$MAPERR              ; map to RMS error
```

```
                58 8ED0  02D7   818          POPL    R8                      ; restore SWB address to R8
                   05    02DA   819          RSB                             ; return fatal error
                         02DB   820
                51   D4  02DB   821 90$:     CLRL    R1                      ; indicate that no subdirectory was
                CF   11  02DD   822          BRB     65$                     ; found and no fatal error
```

```
                     02DF    824          .SBTTL  MATCH, Compare Directory Specification with Pattern String
                     02DF    825
                     02DF    826 ;++
                     02DF    827 ;
                     02DF    828 ; MATCH:        compare directory specification with pattern string
                     02DF    829 ;
                     02DF    830 ;       this routine is responsible for comparing the current directory
                     02DF    831 ;       specification with the current pattern string. it makes this comparison
                     02DF    832 ;       one token/pattern at a time, recursively calling itself with the
                     02DF    833 ;       remainder of the pattern string and/or directory specification when it
                     02DF    834 ;       finds the current token matches the current directory name. the
                     02DF    835 ;       algorithm employed will try all possible matching combinations when
                     02DF    836 ;       ellipses are involved inorder to attempt to find a suitable
                     02DF    837 ;       configuration. note that it is in the very nature of the directory
                     02DF    838 ;       wildcarding algorithm, that this routine can fail only when we
                     02DF    839 ;       have traversed down too deeply, because of an ellipsis, such that the
                     02DF    840 ;       input pattern can never be matched. most often, a status of total
                     02DF    841 ;       or partial success is returned, and in the latter case, information
                     02DF    842 ;       about the next directory to look for is returned as well.
                     02DF    843 ;
                     02DF    844 ; Inputs:
                     02DF    845 ;
                     02DF    846 ;       r6  = descriptor index of current directory name
                     02DF    847 ;       r8  = swb
                     02DF    848 ;       r10 = fwa
                     02DF    849 ;
                     02DF    850 ; outputs:
                     02DF    851 ;
                     02DF    852 ;       r0 = true if match was successful, false for fatal
                     02DF    853 ;       r1 = degree of success - total or partial
                     02DF    854 ;            meaningless if r0 = false
                     02DF    855 ;
                     02DF    856 ;       FWA$B_DIRWCFLGS is set appropriately if the match was totally
                     02DF    857 ;       successful, trashed otherwise
                     02DF    858 ;
                     02DF    859 ;       SWB$L_SCRATCH_PAT contains information about the next directory to search fo
                     02DF    860 ;       when a status of partial success is returned
                     02DF    861 ;
                     02DF    862 ;--
                     02DF    863
 52   0130 CA46   7D 02DF    864 MATCH:  MOVQ    FWA$Q_DIR1(R10)[R6],R2  ; get current directory's descriptor
                     02E5    865
                     02E5    866 ;
                     02E5    867 ; if there are tokens remaining but no directories, this means that not
                     02E5    868 ; enough of the tree has been downward traversed to match the input pattern
                     02E5    869 ; string. a status of partial success is returned so that the downward
                     02E5    870 ; traversal will continue as is required. there is one exception. if the
                     02E5    871 ; current token is an unbounded ellipsis (and thus the last token), it
                     02E5    872 ; matches the empty current directory, so a status of total success is
                     02E5    873 ; returned.
                     02E5    874 ;
                     02E5    875
   01 AB    95 02E5    876          TSTB    SWB$B_PATLEN(R8)        ; if there are no more tokens left
      68    13 02E8    877          BEQL    70$                     ; then branch to handle that condition
      52    B5 02EA    878          TSTW    R2                      ; if both tokens & directories are left
      0E    12 02EC    879          BNEQ    10$                     ; branch to handle that condition too
      00    E1 02EE    880          BBC     #SWB$V_ELLIPSIS,-       ; if the current token is an unbounded
```

```
                07 68       02F0   881                     (R8),1$                         ; ellipsis then return success, else
          03 68    01   F0  02F2   882             BBS     #SWB$V_BOUNDED,(R8),1$          ; return partial success as the current
                0097   31  02F6   883             BRW     120$                            ; directory specification is too short
                0079   31  02F9   884 1$:         BRW     110$
                           02FC   885
                           02FC   886 ;
                           02FC   887 ; if there are both tokens and directories remaining, and the current token
                           02FC   888 ; is an unbounded ellipsis (and thus the last token), we match the current
                           02FC   889 ; directory with the ellipsis, move onto the next directory, and recurse.
                           02FC   890 ; matching the directory with the ellipsis requires adjustment of the
                           02FC   891 ; directory wild card context.
                           02FC   892 ;
                           02FC   893
                      00   E1  02FC   894 10$:         BBC     #SWB$V_ELLIPSIS,-               ; if the current token is not an
                   0D 68       02FE   895                     (R8),30$                        ; ellipsis branch, but if it is an
          64 68    01   E1  0300   896             BBC     #SWB$V_BOUNDED,(R8),95$         ; unbounded ellipsis, then match the
                           0304   897                                                     ; current directory against the
                           0304   898                                                     ; ellipsis, and go continue
```

```
                                0304   900 ;
                                0304   901 ; if there are both tokens and directories remaining, and the current token
                                0304   902 ; is not a bounded ellipsis, then the action take depends upon the nature of
                                0304   903 ; the token. if the token is a bounded ellipsis, the current directory name
                                0304   904 ; is matched against the ellipsis bounds. if the match succeeds, we recurse
                                0304   905 ; with the next token and directory. if the match fails, and it is impossible
                                0304   906 ; to ever match the entire input pattern from the current directory depth, we
                                0304   907 ; return failure, otherwise, we match the current directory with the ellipsis
                                0304   908 ; and recurse with the next directory name. finally, if the current token is
                                0304   909 ; not an ellipsis, we match it directly with the current directory name.
                                0304   910 ; success is handled as above. failure means that we must have prematurely
                                0304   911 ; matched an ellipsis bounds. our course of action is to back up to the last
                                0304   912 ; ellipsis, match the bounds directory to the ellipsis, and continue the
                                0304   913 ; recursion.
                                0304   914 ;
                                0304   915
                     7E    52   7D   0304   916 20$:     MOVQ    R2,-(SP)                ; save the current directory descriptor
                        00CA   30   0307   917           BSBW    NEXT_PATTERN            ; roll forward to the next token
                     52    8E   7D   030A   918           MOVQ    (SP)+,R2                ; restore the directory descriptor
                                0030D  919
                   54    01 A8   9A   030D   920 30$:     MOVZBL  SWB$B_PATLEN(R8),R4     ; create a descriptor for the current
                   55    02 A8   9A   0311   921           MOVZBL  SWB$B_PPOS(R8),R5       ; token by moving its length into r4
                   55    10 A8   C0   0315   922           ADDL2   SWB$Q_PATTERN+4(R8),R5  ; and computing its address in r5
                52   FFFF0000 8F   CA   0319   923           BICL2   #^XFFFF0000,R2          ; clear any stale FSCB bits
                     00000000'EF   16   0320   924           JSB     FMG$MATCH_NAME          ; test for token-directory name match
                           05 50   E9   0326   925           BLBC    R0,40$                  ; branch if no match
                            00A8   30   0329   926           BSBW    NEXT_PATTERN            ; if they match, roll forward to next
                              3C   11   032C   927           BRB     100$                    ; token, and go recurse once more
                                0032E  928
                            0158   30   032E   929 40$:     BSBW    PREV_PATTERN            ; roll back to previous token and if it
                27 68         00   E1   0331   930           BBC     #SWB$V_ELLIPSIS,(R8),90$; isn't an ellipsis continue rollback
                            005F   30   0335   931           BSBW    SET_WCC                 ; otherwise match directory to ellipsis
                           07    56   91   0338   932           CMPB    R6,#FWA$C_MAXSUBDIR     ; if we are at the maximum depth then
                              13   1E   033B   933           BGEQU   60$                     ; go return failure otherwise go
                        0138 CA46   D5   033D   934           TSTL    FWA$Q_DIR1+8(R10)[R6]   ; if there are no more directories, go
                              26   12   0342   935           BNEQ    100$                    ; test if pattern can ever be matched
                                0344   936                                                   ; continue the recursion
                                0344   937
                           03 A8    83   0344   938 50$:     SUBB3   SWB$B_TOKENS_LEFT(R8),- ; if we are at a directory depth such
                54         05 A8         0347   939                   SWB$B_MAXIMUM(R8),R4    ; that the input pattern can never be
                2E AA         54   91   034A   940           CMPB    R4,FWA$B_DIRLEN(R10)    ; matched, go return failure, otherwise
                              25   1E   034E   941           BGEQU   110$                    ; we go return partial success
                                0350   942
                              50   D4   0350   943 60$:     CLRL    R0                      ; if we are to return failure we clear
                              51   D4   0352   944           CLRL    R1                      ; both general registers r0 and r1
                                   05   0354   945           RSB                             ; and return to recurse upwards
```

```
                     0355   947
                     0355   948 ;
                     0355   949 ; if there are neither tokens nor directories left, then the directory
                     0355   950 ; specification does match the pattern, and so we go return total success.
                     0355   951 ; if there are no tokens left, but there are directories, this means we
                     0355   952 ; must have prematurely matched an ellipsis bounds. our course of action
                     0355   953 ; is to backup to the last ellipsis, match the bounds directory to the
                     0355   954 ; ellipsis, and continue the recursion.
                     0355   955 ;
                     0355   956
          52   D5    0355   957 70$:    TSTL    R2                      ; if there are not any tokens or
          37   13    0357   958         BEQL    120$                    ; directories then go return success
        012D   30    0359   959         BSBW    PREV_PATTERN            ; otherwise backup one token
                     035C   960
                     035C   961 ;
                     035C   962 ; when a rollback to the last encountered ellipsis is specified, the
                     035C   963 ; directory matching the ellipsis's bounds is now matched against the ellipsis
                     035C   964 ; itself, the wild card directory context is appropriately altered, and we
                     035C   965 ; attempt to continue the matching process from this position.
                     035C   966 ;
                     035C   967
          012A   30  035C   968 90$:    BSBW    PREV_PATTERN            ; now rollback one token and one
            56   D7  035F   969         DECL    R6                      ; directory, unless can't back up any,
         EC 50   E9  0361   970         BLBC    R0,60$                  ; and if token is not an
      F4 68 00   E1  0364   971         BBC     #SWB$V_ELLIPSIS,(R8),90$; ellipsis continue rollback, otherwise
            2D   10  0368   972 95$:    BSBB    SET_WCC                 ; match dir to ellipsis and continue
                     036A   973
                     036A   974 ;
                     036A   975 ; we want to see if the remaining directory specification matches the
                     036A   976 ; remaining pattern tokens. Thus, we increment the directory descriptor index
                     036A   977 ; to the descriptor of the next directory, and we recurse by having match
                     036A   978 ; call match.
                     036A   979 ;
                     036A   980
    07      56   91  036A   981 100$:   CMPB    R6,#FWA$C_MAXSUBDIR     ; if we are the maximum directory level
            21   13  036D   982         BEQL    120$                    ; then go return total success
            56   D6  036F   983         INCL    R6                      ; increment dir index to next position
          FF6B   30  0371   984         BSBW    MATCH                   ; see if remaining dir & tokens match
                     0374   985         RSB                             ; return final result
```

```
                     0375   987
                     0375   988 ;
                     0375   989 ; if a status of partial success is to be returned, general register r0 is
                     0375   990 ; set to one, general register r1 is cleared, the information about the next
                     0375   991 ; directory to search for is place in the temporary buffer SWB$L_SCRATCH_PAT,
                     0375   992 ; and we return to initiate upwards recursion.
                     0375   993 ;
                     0375   994
      51    D4       0375   995 110$:   CLRL    R1                              ; partial success requires a zeroed r1
14 A8   68    D0     0377   996         MOVL    (R8),SWB$L_SCRATCH_PAT(R8)      ; put token characteristics into buf
        06    E1     037B   997         BBC     #SWB$V_ELLIPSIS_EXISTS,-        ; if no ellipsis exists in the input
           14 68      037D   998                 (R8),130$                       ; pattern go return partial success
        00    E0     037F   999         BBS     #SWB$V_ELLIPSIS,-               ; if the current token is an ellipsis
           10 68      0381  1000                 (R8),130$                       ; go return partial success
        2E AA   91    0383  1001         CMPB    FWA$B_DIRLEN(R10),-             ; if there there are ellipses but they
        06 A8         0386  1002                 SWB$B_FIRST_E(R8)               ; all follow the current token then
        09    1F      0388  1003         BLSSU   130$                            ; go return partial success
              88      038A  1004         BISB2   #SWB$M_ELLIPSIS!-               ; otherwise set the ellipsis and wild
                      038B  1005                 SWB$M_WILD,-                    ; bits in the returned buffer so that
14 A8   05            038B  1006                 SWB$L_SCRATCH_PAT(R8)           ; *.dir;1 will be searched for and
        03    11      038E  1007         BRB     130$                            ; go return partial success
                     0390  1008
                     0390  1009 ;
                     0390  1010 ; if a status of total success is to be returned, both general registers r0
                     0390  1011 ; and r1 are set to one before we return to initiate upwards recursion.
                     0390  1012 ;
                     0390  1013
   51   01    D0     0390  1014 120$:   MOVL    #1,R1                           ; for total success, set both
   50   01    D0     0393  1015 130$:   MOVL    #1,R0                           ; general registers r0 and r1
              05     0396  1016         RSB                                     ; before returning
```

```
                          0397  1018          .SBTTL  SET_WCC, Maintain Directory Wildcard Context
                          0397  1019
                          0397  1020 ;++
                          0397  1021 ;
                          0397  1022 ; SET_WCC:      maintain directory wildcard context
                          0397  1023 ;
                          0397  1024 ;       this routine adjusts the directory wild card context so that it is
                          0397  1025 ;       consistant with the current directory specification. it is only
                          0397  1026 ;       called when a directory name is matched to an ellipsis becomming
                          0397  1027 ;       wild, and requiring that the appropriate bit (corresponding to its
                          0397  1028 ;       level(-1) be set within this field.
                          0397  1029 ;
                          0397  1030 ; inputs:
                          0397  1031 ;
                          0397  1032 ;       r6  = number of bit to be set after shifting subfield
                          0397  1033 ;       r10 = fwa
                          0397  1034 ;
                          0397  1035 ; outputs:
                          0397  1036 ;
                          0397  1037 ;       FWA$B_DIRWCFLGS with a context appropriate for the current
                          0397  1038 ;       directory specification
                          0397  1039 ;
                          0397  1040 ;--
                          0397  1041
                          0397  1042 SET_WCC:
          50    08   56   C3  0397  1043          SUBL3   R6,#FWA$C_MAXSUBDIR+1,R0; get length of subfield to be shifted
51  05 AA   50   56   EF  039B  1044          EXTZV   R6,R0,-                 ; extract the FWA$B_DIRWCFLGS subfield
                          03A1  1045                  FWA$B_DIRWCFLGS(R10),R1 ; to be shifted to the left, shift the
          51   51   01   9C  03A1  1046          ROTL    #1,R1,R1                ; subfield one bit to the left,
05 AA   50   56   51   F0  03A5  1047          INSV    R1,R6,R0,-              ; reinsert the shifted subfield, and
                          03AB  1048                  FWA$B_DIRWCFLGS(R10)    ; set the bit corresponding to the
                          03AB  1049          SSB     R6,FWA$B_DIRWCFLGS(R10) ; directory now matching the ellipsis
                     05   03B0  1050          RSB                             ; return
```

```
                        03B1  1052          .SBTTL  CHK_MFD, Check Current Token to Match MFD
                        03B1  1053
                        03B1  1054 ;++
                        03B1  1055 ;
                        03B1  1056 ; CHK_MFD:      check for mfd
                        03B1  1057 ;
                        03B1  1058 ;       this routine checks the current token to see if it is the mfd
                        03B1  1059 ;
                        03B1  1060 ; inputs:
                        03B1  1061 ;
                        03B1  1062 ;       r4/r5  = descriptor of current token
                        03B1  1063 ;       r10 = fwa, particularly fwa$t_fibbuf
                        03B1  1064 ;
                        03B1  1065 ; outputs:
                        03B1  1066 ;
                        03B1  1067 ;       Z-bit = set iff mfd
                        03B1  1068 ;
                        03B1  1069 ;--
                        03B1  1070
                        03B1  1071 MFD_FID:
           00040004     03B1  1072          .LONG   <FID$C_MFD@16>+FID$C_MFD ; fid of the mfd
                        03B5  1073 MFD_NAME:
 30 30 30 30 30 30      03B5  1074          .ASCII  \000000\                 ; ascii name of mfd
                        03B8  1075
                        03B8  1076 CHK_MFD:
       F3 AF    D1      03B8  1077          CMPL    B^MFD_FID,-              ; if the base directory (i.e. the
    01FE CA             03BE  1078                  FWA$T_FIBBUF+-           ; directory whose file ID is in the DID
                        03C1  1079                  FIB$W_DID(R10)           ; of the FIB) is not the MFD, then
          10    12      03C1  1080          BNEQ    10$                      ; can't be the mfd
       54   06    B1      03C3  1081          CMPW    #6,R4                    ; if the file does not have exactly six
          0B    12      03C6  1082          BNEQ    10$                      ;  chars then it can't be 000000.dir;1
    65    EA AF    D1      03C8  1083          CMPL    B^MFD_NAME,(R5)          ; if the directory isn't 000000.dir;1
          05    12      03CC  1084          BNEQ    10$                      ; then continue the downward traversal
 04 A5    E4 AF    B1      03CE  1085          CMPW    B^MFD_NAME,4(R5)         ;
                        05  03D3  1086 10$:    RSB                              ; return
```

```
                     03D4  1088
                     03D4  1089          .SBTTL  NEXT_PATTERN, Skip to Next Pattern Token
                     03D4  1090
                     03D4  1091 ;++
                     03D4  1092 ;
                     03D4  1093 ; NEXT_PATTERN: skip to next pattern token
                     03D4  1094 ;
                     03D4  1095 ;       skip to the next token in the pattern string.  a
                     03D4  1096 ;       token is defined to be any directory name or an
                     03D4  1097 ;       ellipsis. after skipping to the next token, it
                     03D4  1098 ;       is parsed.
                     03D4  1099 ;
                     03D4  1100 ; inputs:
                     03D4  1101 ;
                     03D4  1102 ;       r8 = swb address
                     03D4  1103 ;
                     03D4  1104 ; outputs:
                     03D4  1105 ;
                     03D4  1106 ;--
                     03D4  1107
                     03D4  1108 NEXT_PATTERN:
03 68   00   E0      03D4  1109          BBS     #SWB$V_ELLIPSIS,(R8),10$; if the current token is not an
      03 A8   97     03D8  1110          DECB    SWB$B_TOKENS_LEFT(R8)   ; ellipsis, decrement the token counter
                     03DB  1111
      01 A8   80     03DB  1112 10$:     ADDB2   SWB$B_PATLEN(R8),-      ; position the pattern offset
      02 A8          03DE  1113                  SWB$B_PPOS(R8)          ; to the next delimiter
      02 A8   91     03E0  1114          CMPB    SWB$B_PPOS(R8),-        ; if there are no more tokens (i.e. we
      0C A8          03E3  1115                  SWB$Q_PATTERN(R8)       ;  have reached the end of the token
         07   1E     03E5  1116          BGEQU   PARSE_PATTERN           ;  string) or if the next token is an
         03   E1     03E7  1117          BBC     #SWB$V_DELIMITER,-      ; ellipsis, then there is no
      03 68          03E9  1118                  (R8),PARSE_PATTERN      ; need to bypass the delimiter token
      02 A8   96     03EB  1119          INCB    SWB$B_PPOS(R8)          ; too, otherwise, there is such a need
                     03EE  1120                                          ; drop thru to parse_pattern
```

```
                     03EE  1122          .SBTTL  PARSE_PATTERN, Parse Current Pattern Token
                     03EE  1123
                     03EE  1124 ;++
                     03EE  1125 ;
                     03EE  1126 ; PARSE_PATTERN: parse current pattern token
                     03EE  1127 ;
                     03EE  1128 ;       set the length and characteristics flags of the
                     03EE  1129 ;       current token in the pattern string.
                     03EE  1130 ;
                     03EE  1131 ; inputs:
                     03EE  1132 ;
                     03EE  1133 ;       r8 = swb address
                     03EE  1134 ;
                     03EE  1135 ; outputs:
                     03EE  1136 ;
                     03EE  1137 ;       swb$b_patlen = length of token, 0 if none
                     03EE  1138 ;       swb$b_flags  = flags describing token
                     03EE  1139 ;
                     03EE  1140 ;--
                     03EE  1141
                     03EE  1142 PARSE_PATTERN:
        68    0F  8A 03EE  1143          BICB2   #SWB$M_ELLIPSIS!-       ; clear status flags
                     03F1  1144                  SWB$M_BOUNDED!-
                     03F1  1145                  SWB$M_WILD!-
                     03F1  1146                  SWB$M_DELIMITER,(R8)
     52    0C A8  7D 03F1  1147          MOVQ    SWB$Q_PATTERN(R8),R2    ; get string descriptor
     51    02 A8  9A 03F5  1148          MOVZBL  SWB$B_PPOS(R8),R1       ; current offset into string
        53    51  C0 03F9  1149          ADDL2   R1,R3                   ; address of string left
        52    51  C2 03FC  1150          SUBL2   R1,R2                   ; length of string left
              4B  13 03FF  1151          BEQL    50$                     ; branch if nothing left
                     0401  1152
                     0401  1153 ;
                     0401  1154 ; check if token is an ellipsis
                     0401  1155 ;
                     0401  1156
        03    52    D1 0401  1157 20$:    CMPL    R2,#3                   ; 3 characters left?
              18  1F 0404  1158          BLSSU   10$                     ; branch if not
  2E2E 8F     63  B1 0406  1159          CMPW    (R3),#^A'..'            ; ellipsis?
              11  12 040B  1160          BNEQ    10$                     ; branch if not
        68    07  88 040D  1161          BISB2   #SWB$M_ELLIPSIS!-       ; mark ellipsis is current
                     0410  1162                  SWB$M_BOUNDED!-
                     0410  1163                  SWB$M_WILD,(R8)
        03    52    D1 0410  1164          CMPL    R2,#3                   ; anything following ellipsis?
              04  1A 0413  1165          BGTRU   5$                      ; branch if yes
                     0415  1166          CSB     #SWB$V_BOUNDED,(R8)     ; mark unbounded
        52    03    D0 0419  1167 5$:     MOVL    #3,R2                   ; set length of ellipsis
              2E  11 041C  1168          BRB     50$
                     041E  1169
                     041E  1170 ;
                     041E  1171 ; find delimiter following token
                     041E  1172 ;
                     041E  1173
  63    52    2E  3A 041E  1174 10$:    LOCC    #^A'.',R2,(R3)          ; find next dot in string
              15  13 0422  1175          BEQL    30$                     ; branch if not found
                     0424  1176          SSB     #SWB$V_DELIMITER,(R8)   ; assume 1 char, delimiter following
        03    50    D1 0428  1177          CMPL    R0,#3                   ; if there are 3 characters left
              0C  19 042B  1178          BLSS    30$
```

```
                2E2E 8F    01 A1    B1  0420  1179          CMPW    1(R1),#^A'..'           ; then check for ellipsis following
                           04 12    0433  1180          BNEQ    30$                     ; branch if not
                                    0435  1181          CSB     #SWB$V_DELIMITER,(R8)   ; if so, set no delimiter after token
                                    0439  1182
                                    0439  1183 ;
                                    0439  1184 ; if token has wild characters, set wild flag
                                    0439  1185 ;
                                    0439  1186
                     52    50 C2    0439  1187 30$:    SUBL2   R0,R2                   ; length of token passed by
                63   52    2A 3A    043C  1188          LOCC    #^A'*',R2,(R3)          ; search token for wild *
                           06 12    0440  1189          BNEQ    40$                     ; if found, set bit
                63   52    25 3A    0442  1190          LOCC    #^A'%',R2,(R3)          ; search token for wild %
                           04 13    0446  1191          BEQL    50$                     ; if not found, set length
                                    0448  1192 40$:    SSB     #SWB$V_WILD,(R8)        ; mark token has wild characters
                                    044C  1193
                                    044C  1194 ;
                                    044C  1195 ; set length of new token in string (r2 = length)
                                    044C  1196 ;
                                    044C  1197
                   01 A8    52 90    044C  1198 50$:    MOVB    R2,SWB$B_PATLEN(R8)     ; set token length
                                    05 0450  1199          RSB                             ; return
```

```
                     0451  1201          .SBTTL  PREV_DIR, Strip one Directory Name from Directory Specification
                     0451  1202
                     0451  1203 ;++
                     0451  1204 ;
                     0451  1205 ; PREV_DIR: strip one directory name from directory specification
                     0451  1206 ;
                     0451  1207 ;       strip the last directory name from the current
                     0451  1208 ;       directory specification being built.
                     0451  1209 ;
                     0451  1210 ; inputs:
                     0451  1211 ;
                     0451  1212 ;       r10 = fwa address
                     0451  1213 ;
                     0451  1214 ; outputs:
                     0451  1215 ;
                     0451  1216 ;       r4/r5 = descriptor of stripped directory name
                     0451  1217 ;       r0    = false if no names left to strip, else true
                     0451  1218 ;
                     0451  1219 ;--
                     0451  1220
                     0451  1221 PREV_DIR:
  16 6A    1B   E1   0451  1222          BBC     #FWA$V_GRPMBR,(R10),10$ ; branch if not uic format
        14    10   0455  1223          BSBB    10$                     ; strip member portion
      2C 50   E9   0457  1224          BLBC    R0,90$                  ; branch if nothing left
50   0130 CA   3C   045A  1225          MOVZWL  FWA$Q_DIR1(R10),R0      ; obtain length of group portion
0130 CA   54   A0   045F  1226          ADDW2   R4,FWA$Q_DIR1(R10)      ; concatenate group and member i.e.
   65   54   28   0464  1227          MOVC3   R4,(R5),-               ; append member onto end of group name
     0134 DA40       0467  1228                  @FWA$Q_DIR1+4(R10)[R0]  ; and then strip the whole thing off
                     046B  1229
  51   2E AA   9A   046B  1230 10$:    MOVZBL  FWA$B_DIRLEN(R10),R1    ; get number of slots in use
        51    D7   046F  1231          DECL    R1                      ; backup over last slot
        13    19   0471  1232          BLSS    90$                     ; branch if none left
   2E AA   51   90   0473  1233          MOVB    R1,FWA$B_DIRLEN(R10)    ; store updated slots in use
51   0130 CA41   7E   0477  1234          MOVAQ   FWA$Q_DIR1(R10)[R1],R1  ; address of stripped descriptor
     54   61   7D   047D  1235          MOVQ    (R1),R4                 ; return descriptor to caller
        61   B4   0480  1236          CLRW    (R1)                    ; zero length of FWA descriptor
     50   01   D0   0482  1237          MOVL    #1,R0                   ; indicate success and
             05   0485  1238          RSB                             ; return
                     0486  1239
        50   D4   0486  1240 90$:    CLRL    R0                      ; return error - no names left
             05   0488  1241          RSB
```

```
                     0489  1243          .SBTTL  PREV_PATTERN, Backup to Previous Pattern Token
                     0489  1244
                     0489  1245 ;++
                     0489  1246 ;
                     0489  1247 ; PREV_PATTERN: backup to previous pattern token
                     0489  1248 ;
                     0489  1249 ;       backup one token in the pattern string.
                     0489  1250 ;
                     0489  1251 ; inputs:
                     0489  1252 ;
                     0489  1253 ;       r8 = swb ad  ess
                     0489  1254 ;
                     0489  1255 ; outputs:
                     0489  1256 ;
                     0489  1257 ;       r0 = false if no tokens left to strip, else true
                     0489  1258 ;--
                     0489  1259
                     0489  1260 PREV_PATTERN:
    52   0C A8  7D   0489  1261          MOVQ    SWB$Q_PATTERN(R8),R2    ; get string descriptor
    51   02 A8  9A   048D  1262          MOVZBL  SWB$B_PPOS(R8),R1       ; current offset into string
           38   13   0491  1263          BEQL    80$                     ; branch if already at start
        03   51   D1   0493  1264          CMPL    R1,#3                   ; if the previous token is less than
           18   19   0496  1265          BLSS    5$                      ; two chars then it can't be an ...
52   FD A341  9E   0498  1266          MOVAB   -3(R3)[R1],R2           ; if the previous token does not have
62   2E2E 8F   B1   049D  1267          CMPW    #^A'..',(R2)            ; two consecutive dots it can't be an
           0C   12   04A2  1268          BNEQ    5$                      ; ellipsis, likewise if it doesn't have
FF A341    2E   91   04A4  1269          CMPB    #^A'.',-1(R3)[R1]       ; three consecutive dots it can't be an
           05   12   04A9  1270          BNEQ    5$                      ; ellipsis, but if previous token is an
        51   04   C2   04AB  1271          SUBL2   #3+1,R1                 ; ellipsis, backup 4 places, and go
           0F   11   04AE  1272          BRB     50$                     ; update the position offset
           51   D7   04B0  1273 5$:       DECL    R1                      ; if so, skip over it
           51   D7   04B2  1274 10$:      DECL    R1                      ; skip to previous character
           09   19   04B4  1275          BLSS    50$                     ; branch if at start of string
        2E   6341   91   04B6  1276          CMPB    (R3)[R1],#^A'.'         ; reached a dot?
           F6   12   04BA  1277          BNEQ    10$                     ; branch if not
        03 A8   96   04BC  1278          INCB    SWB$B_TOKENS_LEFT(R8)   ; increment nonellipsis token counter
02 A8   51   01   81   04BF  1279 50$:      ADDB3   #1,R1,SWB$B_PPOS(R8)    ; store updated offset
           FF27   30   04C4  1280          BSBW    PARSE_PATTERN           ; parse current pattern token
        50   01   D0   04C7  1281          MOVL    #1,R0                   ; indicate success and
                05   04CA  1282          RSB                             ; return
           50   D4   04CB  1283 80$:      CLRL    R0                      ; indicate failure and
                05   04CD  1284          RSB                             ; return
```

```
                        04CE  1286          .SBTTL  APPEND_DIR, Append Directory to end of Directory Specification
                        04CE  1287
                        04CE  1288 ;++
                        04CE  1289 ;
                        04CE  1290 ; APPEND_DIR: append directory to end of directory specification
                        04CE  1291 ;
                        04CE  1292 ;       this routine appends a directory name to the end
                        04CE  1293 ;       of the current directory spec being assembled.
                        04CE  1294 ;
                        04CE  1295 ; inputs:
                        04CE  1296 ;
                        04CE  1297 ;       r4/r5 = descriptor of directory name.
                        04CE  1298 ;       r10   = fwa address
                        04CE  1299 ;       r8    = swb address
                        04CE  1300 ;
                        04CE  1301 ; outputs:
                        04CE  1302 ;
                        04CE  1303 ;       r1-r3 destroyed
                        04CE  1304 ;--
                        04CE  1305
                        04CE  1306 APPEND_DIR:
     08 6A    1B   E1   04CE  1307          BBC     #FWA$V_GRPMBR,(R10),10$ ; branch if not uic format
        54    03   D0   04D2  1308          MOVL    #3,R4                   ; only 3 characters in each part
              03   10   04D5  1309          BSBB    10$                     ; append group portion
        55    54   C0   04D7  1310          ADDL2   R4,R5                   ; skip to member portion of string
                         04DA  1311                                          ; and append it by falling thru
     50    2E AA   9A   04DA  1312 10$:     MOVZBL  FWA$B_DIRLEN(R10),R0    ; get number of names in use
51    0130 CA40   7E   04DE  1313          MOVAQ   FWA$Q_DIR1(R10)[R0],R1  ; address of next slot descriptor
        61    54   B0   04E4  1314          MOVW    R4,(R1)                 ; set length in descriptor
     50    04 A1   D0   04E7  1315          MOVL    4(R1),R0                ; get address of buffer from descriptor
        7E    54   7D   04EB  1316          MOVQ    R4,-(SP)                ; save input descriptor
  60    65    54   28   04EE  1317          MOVC3   R4,(R5),(R0)            ; move string into buffer
        54    8E   7D   04F2  1318          MOVQ    (SP)+,R4                ; restore input descriptor
           2E AA   96   04F5  1319          INCB    FWA$B_DIRLEN(R10)       ; increment names in use
                   05   04F8  1320          RSB                             ; return
```

```
                     04F9  1322          .SBTTL  SET_BASE, FIND Base Directory and Determine its Position
                     04F9  1323
                     04F9  1324 ;++
                     04F9  1325 ;
                     04F9  1326 ; SET_BASE:     find base directory and determine its position
                     04F9  1327 ;
                     04F9  1328 ;       the base directory will either be the MFD, if the very first pattern
                     04F9  1329 ;       token is wild, or the last nonwild name after finding either the first
                     04F9  1330 ;       wild token in the pattern string, or the end of the pattern string.
                     04F9  1331 ;       descriptors are created in the FWA, for each of the leading nonwild
                     04F9  1332 ;       names encountered up to and including the base directory. the DID of
                     04F9  1333 ;       the base directory is then set in the FIB. upon return, the current
                     04F9  1334 ;       token will be the first wild token encountered, or the end of the
                     04F9  1335 ;       pattern string if all tokens encountered were nonwild.
                     04F9  1336 ;
                     04F9  1337 ; inputs:
                     04F9  1338 ;
                     04F9  1339 ;       r8  = swb address
                     04F9  1340 ;       r10 = fwa address
                     04F9  1341 ;
                     04F9  1342 ;
                     04F9  1343 ; outputs:
                     04F9  1344 ;
                     04F9  1345 ;       r0 = true if base directory DID has been found
                     04F9  1346 ;            false otherwise
                     04F9  1347 ;--
                     04F9  1348
                     04F9  1349 SET_BASE:
        68    02   E0  04F9  1350          BBS     #SWB$V_WILD,(R8),-      ; if token is wild, then find DID
                 16       04FC  1351                  FIND_DIR
    54    01   A8   9A  04FD  1352          MOVZBL  SWB$B_PATLEN(R8),R4     ; get length of current token
    55    02   A8   9A  0501  1353          MOVZBL  SWB$B_PPOS(R8),R5       ; get offset to current token
    55    10   A8   C0  0505  1354          ADDL2   SWB$Q_PATTERN+4(R8),R5  ; compute address of current token
             C3    10  0509  1355          BSBB    APPEND_DIR              ; append token to directory spec
           FEC6    30  050B  1356          BSBW    NEXT_PATTERN            ; skip to next pattern token
        01   A8   95  050E  1357          TSTB    SWB$B_PATLEN(R8)        ; continue as long as there are
             E6    12  0511  1358          BNEQ    SET_BASE                ; nonwild tokens to copy
                       0513  1359                                          ; drop through to FIND_DIR
```

```
                     0513  1361          .SBTTL  FIND_DIR, Determine Current Directory Position
                     0513  1362
                     0513  1363 ;++
                     0513  1364 ;
                     0513  1365 ; FIND_DIR
                     0513  1366 ;
                     0513  1367 ;       Determine current directory position
                     0513  1368 ;
                     0513  1369 ;       Determine the did of the current result directory
                     0513  1370 ;       specification.
                     0513  1371 ;
                     0513  1372 ; inputs:
                     0513  1373 ;
                     0513  1374 ;       r8  = swb address
                     0513  1375 ;       r9  = ifab address
                     0513  1376 ;       r10 = fwa address
                     0513  1377 ;
                     0513  1378 ; outputs:
                     0513  1379 ;
                     0513  1380 ;       r0 = status
                     0513  1381 ;       the did in the fib is set.
                     0513  1382 ;
                     0513  1383 ;       registers r4,r5,r8 are saved.
                     0513  1384 ;--
                     0513  1385 ;
                     0513  1386
                     0513  1387 FIND_DIR:
    2E AA    95      0513  1388          TSTB    FWA$B_DIRLEN(R10)       ; if the directory specification is
       22    12      0516  1389          BNEQ    20$                     ; not empty then setup to find the DID
6A     3A    E0      0518  1390          BBS     #FWA$V_ROOT_DIR,(R10),- ; if there was a root directory then
       0D            051B  1391                  10$                     ;  get the DID the hard way
FE91   CF    D0      051C  1392          MOVL    W^MFD_FID,-             ; if the directory specification is
01FE   CA            0520  1393                  FWA$T_FIBBUF+-          ;  empty then the DID defaults to the
                     0523  1394                  FIB$W_DID(R10)          ;  DID of the MFD
0202   CA    B4      0523  1395          CLRW    FWA$T_FIBBUF+-          ;
                     0527  1396                  FIB$W_DID_RVN(R10)      ;
       33    11      0527  1397          BRB     40$                     ; go return success
                     0529  1398
                     0529  1399 ;
                     0529  1400 ; In order to get the DID of the rooted MFD we force SETDID to stop searching
                     0529  1401 ; directories by clearing the length of the first normal directory descriptor
                     0529  1402 ;
                     0529  1403
0130   CA    DD      0529  1404 10$:     PUSHL   FWA$Q_DIR1(R10)         ; save it the length
0130   CA    D4      052D  1405          CLRL    FWA$Q_DIR1(R10)         ; clear it
       07    10      0531  1406          BSBB    20$                     ; call SETDID to find the DID
0130   CA  8ED0      0533  1407          POPL    FWA$Q_DIR1(R10)         ; restore the length
       1F    11      0538  1408          BRB     30$                     ; exit
                     053A  1409
0130   8F    BB      053A  1410 20$:     PUSHR   #^M<R4,R5,R8>           ; save registers R4-R8 and the wildcard
0204   CA    DD      053E  1411          PUSHL   FWA$T_FIBBUF+-          ; context over DID lookup by pushing
                     0542  1412                  FIB$L_WCC(R10)          ; them on the current stack
  58   24    A9      0542  1413          MOVL    IFB$L_LAST_FAB(R9),R8   ; obtain the address of the FAB
00000000'EF  16      0546  1414          JSB     RM$SETDID_ALT           ; set DID in the FIB
0204   CA  8ED0      054C  1415          POPL    FWA$T_FIBBUF+-          ; restore registers R4-R8 and the
                     0551  1416                  FIB$L_WCC(R10)          ; wildcard context, saved over DID
0130   8F    BA      0551  1417          POPR    #^M<R4,R5,R8>           ; lookup, from the current stack
```

```
                04   E0  0555  1418          BBS     #IMP$V_IORUNDOWN,-      ; if I/O Rundown is in progress,
             08 6B       0557  1419                  (R11),TOO$              ; then RMOWILD is prematurely aborted
             07 50   E9  0559  1420  30$:     BLBC    R0,90$                  ; branch if any error
                         055C  1421  40$:     SSB     #SWB$V_VALID_DID,(R8)   ; set the valid DID bit in the SWB
          50    01   D0  0560  1422          MOVL    #1,R0                   ; otherwise indicate success
                05   0563  1423  90$:     RSB                             ; and return
                         0564  1424
       58    24 A9   D0  0564  1425  100$:   MOVL    IFB$L_LAST_FAB(R9),R8   ; abort by obtaining the address of the  FAB
                         0568  1426          RMSERR  NMF                     ;  setting a default error of no more
                         056D  1427                                          ;  files both in R0 and in the FAB,
                05   056D  1428          RSB                             ;  and returning
                         056E  1429
                         056E  1430          .END
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
APPEND_DIR                        000004CE R      01
CHK_MFD                           000003BB R      01
FAB$L_STV                       = 0000000C
FIB$C_LENGTH                    = 00000040
FIB$L_WCC                       = 00000010
FIB$M_WILD                      = 00000100
FIB$W_DID                       = 0000000A
FIB$W_DID_RVN                   = 0000000E
FIB$W_NMCTL                     = 00000014
FID$C_MFD                       = 00000004
FIND_DIR                          00000513 R      01
FMG$MATCH_NAME                    ******** X    01
FSCB$V_ELIPS                    = 00000010
FWA$B_DIRLEN                    = 0000002E
FWA$B_DIRWCFLGS                 = 00000005
FWA$C_MAXSUBDIR                 = 00000007
FWA$L_SWB_PTR                   = 0000004C
FWA$Q_DIRT                      = 00000130
FWA$Q_FIB                       = 00000010
FWA$Q_NAME                      = 00000170
FWA$Q_RNS                       = 00000188
FWA$S_NAMEBUF                   = 00000100
FWA$S_TYPEBUF                   = 00000028
FWA$S_VERBUF                    = 00000006
FWA$T_FIBBUF                    = 000001F4
FWA$T_NAMEBUF                   = 000004B6
FWA$V_GRPMBR                    = 0000001B
FWA$V_ROOT_DIR                  = 0000003A
FWA$V_WILD_DIR                  = 0000001C
FWA$V_WILD_GRP                  = 00000028
IFB$L_LAST_FAB                  = 00000024
IFB$L_RNS_LEN                   = 0000006C
IMP$V_IORUNDOWN                 = 00000004
IO$_ACCESS                      = 00000032
MATCH                             000002DF R      01
MFD_FID                           000003B1 R      01
MFD_NAME                          000003B5 R      01
NEXT_PATTERN                      000003D4 R      01
NEXT_SUBDIR                       000001F4 R      01
PARSE_PATTERN                     000003EE R      01
PREV_DIR                          00000451 R      01
PREV_PATTERN                      00000489 R      01
RMS$FCPFNC                        ********   X    01
RMS$GETBLK1                       ********   X    01
RMS$INIT_SWB                      00000000 RG     01
RMS$MAPERR                        ********   X    01
RMS$NEXTDIR                       000000CB RG     01
RMS$SETDID_ALT                    ********   X    01
RMS$_FND                        = 0001C02A
RMS$_NMF                        = 000182CA
RMS$_NOVALPRS                   = 0001830A
SET_BASE                          000004F9 R      01
SET_WCC                           00000397 R      01
SS$_BADIRECTORY                 = 00000828
SS$_NOMOREFILES                 = 00000930
SS$_NOSUCHFILE                  = 00000910
SWB$B_BID                       = 00000008
SWB$B_DIRWCFLGS                 = 00000007
SWB$B_FIRST_E                   = 00000006
SWB$B_MAXIMUM                   = 00000005
SWB$B_MINIMUM                   = 00000004
SWB$B_PATLEN                    = 00000001
SWB$B_PPOS                      = 00000002
SWB$B_TOKENS_LEFT               = 00000003
SWB$C_BID                       = 0000002A
SWB$C_BLN                       = 00000148
SWB$L_SCRATCH_PAT               = 00000014
SWB$M_BOUNDED                   = 00000002
SWB$M_DELIMITER                 = 00000008
SWB$M_ELLIPSIS                  = 00000001
SWB$M_WILD                      = 00000004
SWB$Q_PATTERN                   = 0000000C
SWB$T_PATTERN_BUF               = 00000048
SWB$T_SCRATCH_BUF               = 00000018
SWB$V_BOUNDED                   = 00000001
SWB$V_DELIMITER                 = 00000003
SWB$V_ELLIPSIS                  = 00000000
SWB$V_ELLIPSIS_EXISTS           = 00000006
SWB$V_FIRST                     = 00000005
SWB$V_TRAVERSE                  = 00000004
SWB$V_VALID_DID                 = 00000007
SWB$V_WILD                      = 00000002
UIC_EXPAND                        000000B2 R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM$RMSFILENAME	0000056E (1390.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.09	00:00:00.65
Command processing	115	00:00:00.80	00:00:04.45
Pass 1	540	00:00:22.28	00:00:49.62
Symbol table sort	0	00:00:03.47	00:00:05.41
Pass 2	265	00:00:05.45	00:00:12.24
Symbol table output	12	00:00:00.14	00:00:00.48
Psect synopsis output	2	00:00:00.02	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	966	00:00:32.26	00:01:13.07

The working set limit was 1950 pages.
127870 bytes (250 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2295 non-local and 96 local symbols.
1430 source lines were read in Pass 1, producing 16 object records in Pass 2.
27 pages of virtual memory were used to define 26 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	13
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	22

2427 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RMOWILD/OBJ=OBJ$:RMOWILD MSRC$:RMOWILD/UPDATE=(ENH$:RMOWILD)+EXECML$/LIB+LIB$:RMS/LIB
```